Niubility

**ECE 6332**

**2012 Fall**

**Design Review II**

Eric Zhang

Xinfei Guo

1. Summaries of papers

We have done with building the whole circuitry in Cadence design environment. The following papers are good reference for us to optimize the design and analyze the trade offs.

[1] Tripti Sharma, k.G.Sharma, Prof. B.P.Singh, Neha Arora, “High Speed, Low Power 8T Full Adder Cell with 45% Improvement in Threshold Loss Problem”, Recent Advances in Networking, VLSI and Signal Processing.

In this paper, the author proposes a low power full adder cell with least MOS transistor count that reduces the serious problem of threshold loss. In the meantime, it considerably increases the speed. The simulation has been carried out on Tanner EDA tool on BSIM3v3 90nm and 130nm technologies. The result shows the technology independence of the circuit.

The idea of the full adder is to implement 3 transistor XOR gate, which utilize the property of pass gate. We can try to implement the similar adder in our design, especially the adder in 2 bit multiplier. We found this kind of adder saves a lot of power and delay. But it has significant drawbacks, like lack of driving capability. If they are cascaded, their performance degrades significantly. So we have to use carefully.

[2] Parag Kulkarni , Puneet Gupta , Milos Ercegovac, Trading Accuracy for Power with an Underdesigned Multiplier Architecture, Proceedings of the 2011 24th International Conference on VLSI Design, p.346-351, January 02-07, 2011

In this paper, they proposed a similar idea that multiplier can absorb some errors which allows for quality to be traded off for power. Their inaccurate multipliers achieve an average power saving of about 40% over accurate multiplier designs, for an average error of 1.39%-3.32%. They also implement the multiplier in image filtering and JPEG applications and compare it with voltage scaling and bitwidth truncation based method.

We can utilize the idea from this paper that develops the methods to correct error if needed. This will result in some power penalty. Also, what we need to do is to analyze the tradeoffs between power and precision. In this paper, they have a bunch of arguments in accuracy and power. So we can use the same way to make some arguments about this. Lastly, they implement the multiplier in some applications, which will make more sense. For us, we don’t have enough time to do that, but that can be our future work. Ultimately, we hope our multiplier can be used in some image filtering or DSP applications.

|  |  |
| --- | --- |
| [3] | Ali Manzak , Chaitali Chakrabarti, Variable voltage task scheduling algorithms for minimizing energy/power, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.11 n.2, p.270-276, April 2003 |

In this paper, they propose variable voltage task scheduling algorithms that minimize energy or minimize peak power for the case when the task arrival times, deadline times, execution times, periods and switching activities are given. This was a high level voltage scaling description. They theoretically determine the relation between the operating voltages for the minimum energy and minimum peak power assignment using the Lagrange multiplier method and develop heuristics that use this relation.

This paper gives us some theoretical instructions of how we implement voltage scaling in terms of algorithms. This will be helpful for us to know how we can analyze the voltage scaling in high level and it also helps us in understanding tradeoffs between energy/power savings and the complexity of the algorithms.

[4] G. Sathiyabama, Raja Shailaja, A survey of low power high speed full adder, International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 9, September 2012

In this paper various types of full adders design are performed. It can be said as a library of different full adder circuits, where we can explore the trade offs of different adders. It also provides a detailed performance and power consumption comparison of existing adders. This is helpful for our adder implementation.

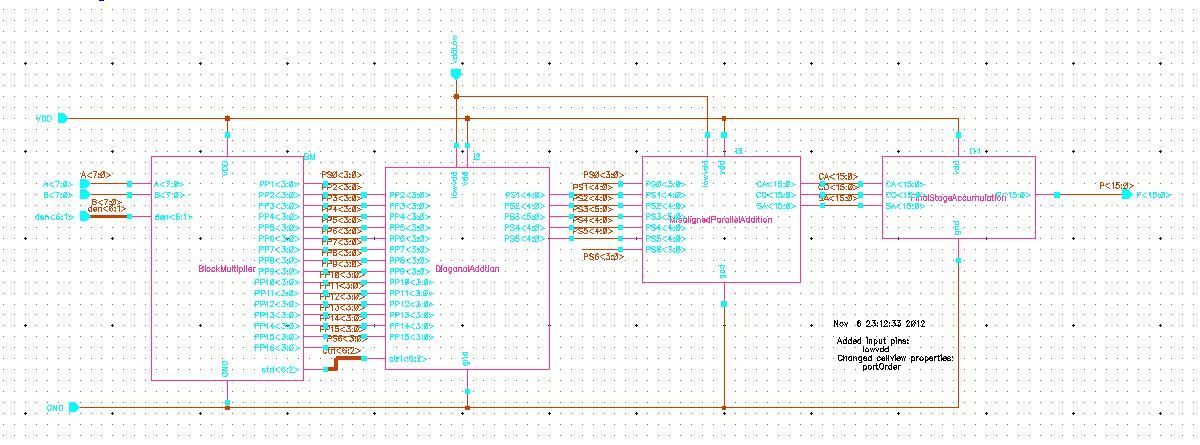
1. Schematics
2. Top level schematic

Fig. 1 Top Level Schematic

Fig.1 is the top level circuit of our multiplier. From left to right, the blocks are block multiplier, Diagonal Additions, Misaligned addition block and final addition block.

1. Block Multipliers

Block multipliers are multiplication of two-bit block. It includes a zero detector in the first stage, which is used to detect the zeros for 2-bit blocks. In this way, we can turn off some corresponding multiplier blocks to save the power. In circuit implementation, zero detector is just several NOR gates.

To control the precision, we have six disable signals which will disable the LSB blocks. So we utilize the idea of coarse power gating to enable and disable the 2bit multiplier. Fig.2 shows the implementation of power gating. We have NMOS transistor connecting to each output to avoid output floating when it is in sleep mode.

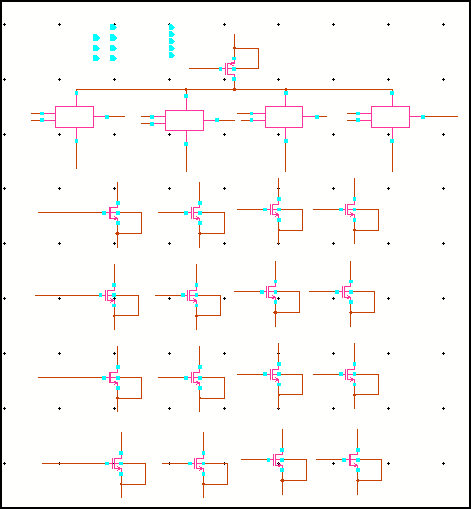


Fig. 2 Implementation of coarse power gating

To pick the size of header NMOS, we used the parameter analysis tool in Cadence ADE. Fig. 3 shows when PMOS is 4 times wider than unit one, we have optimized Et product. So in our design, we pick it 4.

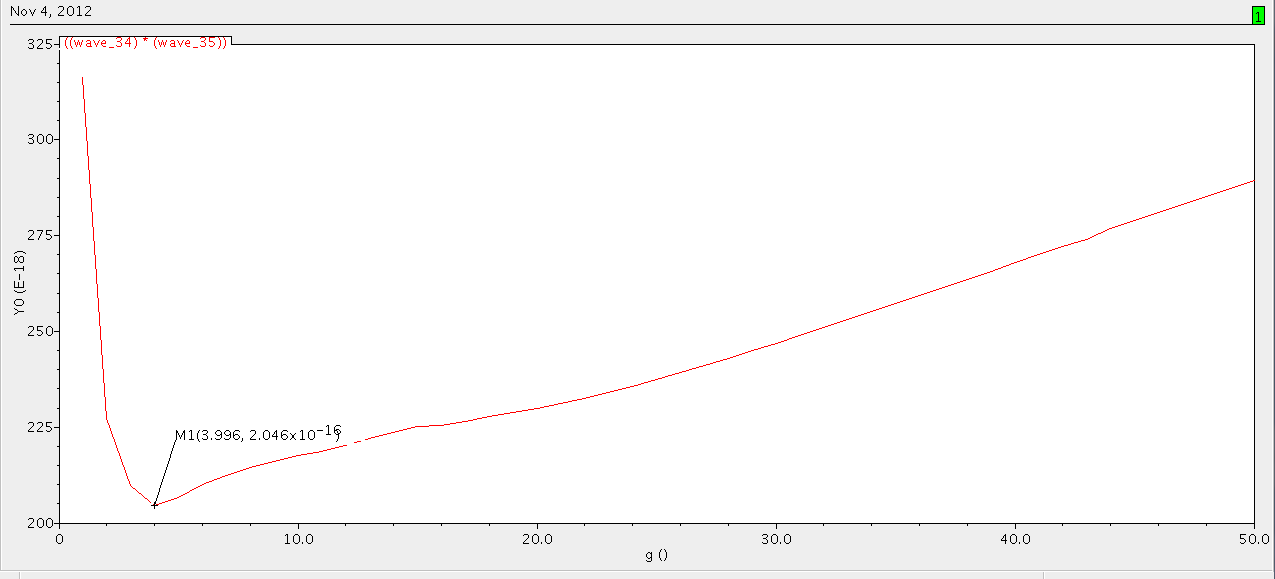


Fig. 3 The width of Header PMOS vs. Et product

Fig. 4 shows the leakage power when it is in sleep mode for different size of Header NMOS. As we can see from the result, the power is fairly small compared with operation mode.

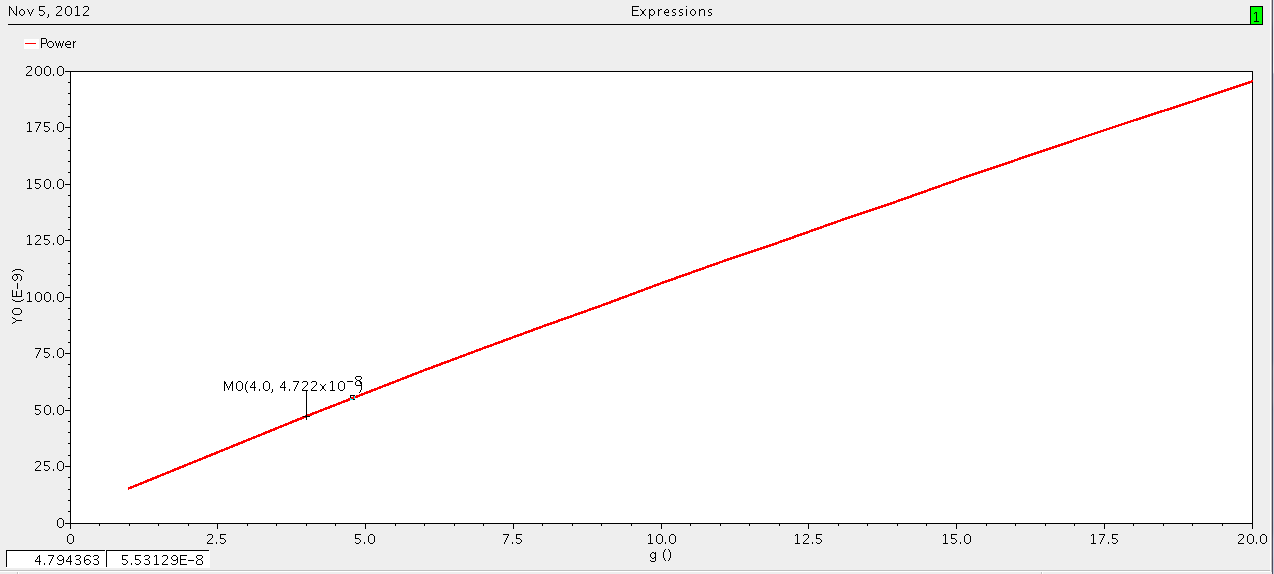


Fig. 4 The width of Header PMOS vs. leakage power

Fig. 5 shows the overall schematic of Block Multiplication.

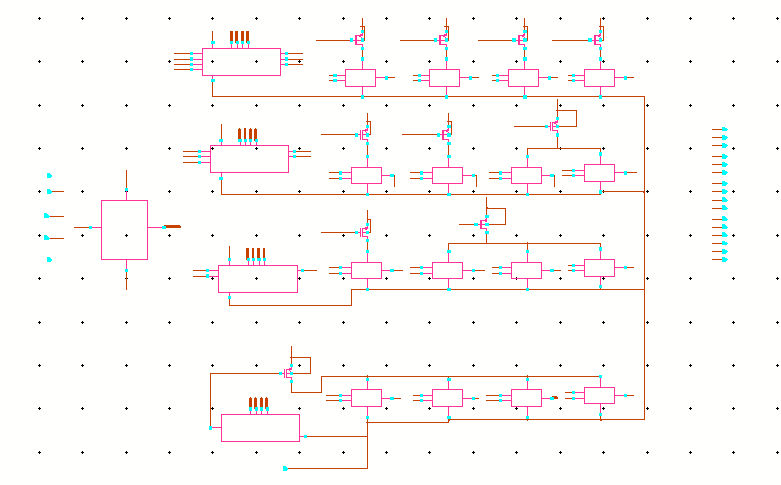


Fig. 5 overall schematic of Block Multiplication

2bit Multiplier

0s

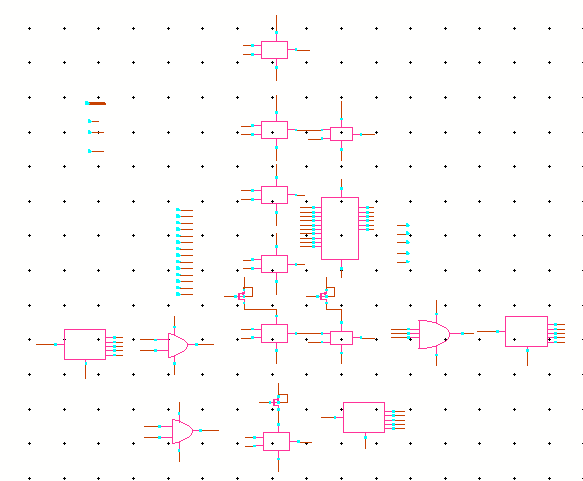
Power Gating

Header PMOS

Footer NMOS

Zero Detector

1. Diagonal Addition



Footer NMOS

4 bit Adder

5 bit Adder

Fig. 6 overall schematic of Diagonal Addition

Fig. 6 shows the overall schematic of diagonal addition. As the block multiplication part, we have the disable signal for block multiplier, so we can still disable the corresponding addition to save more power.

1. Misaligned Addition

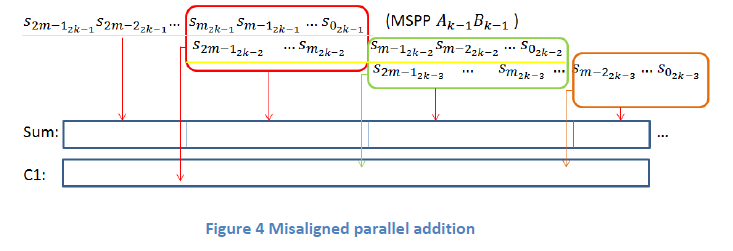
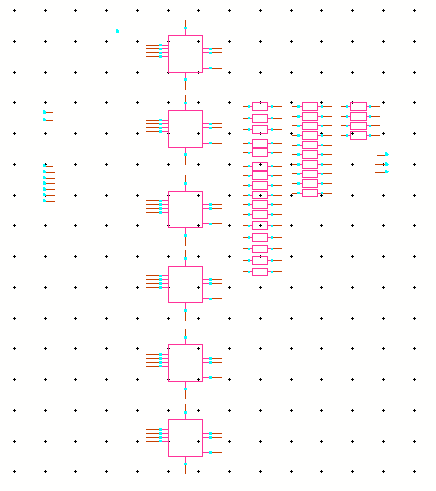
****

Fig. 7 Misaligned Addition description

Shown in fig.7 is misaligned addition illustration. It is how we process the diagonal addition sums. Fig. 8 shows the circuit implementation of misaligned addition.



Buffer

Adder

Fig. 8 Misaligned Addition Circuitry

1. Final Addition Stage

As shown in fig. 9, final addition stage is the last stage to process the sums.

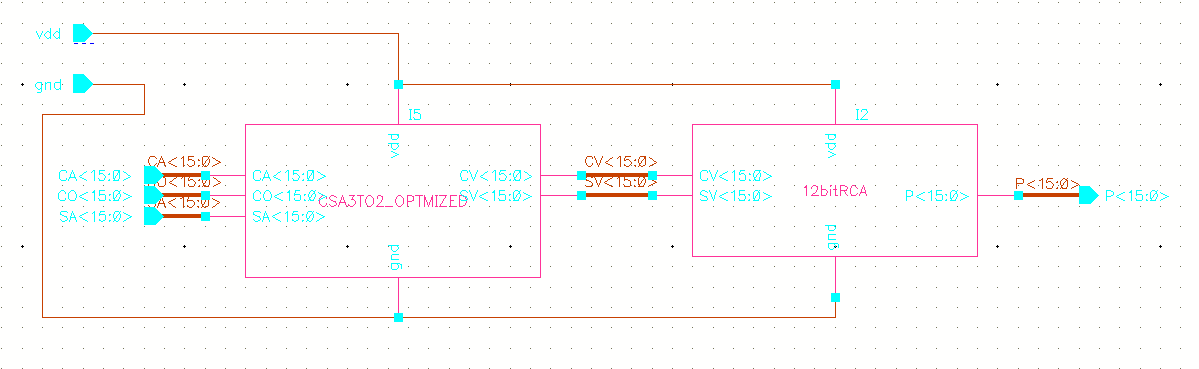


Fig. 9 Final Addition Stage

1. Simulation Results
2. Adder Design

Adder is the most important parts in multiplier, so optimizing the adder can gain huge benefits.

1. Pick the size of Adder

In misaligned addition and final addition stage, we all use carry ripple adder, so the purpose of sizing the adder is to decrease the delay in critical path. Fig. 10 shows the parameter analysis result. From the result, we pick all the PMOS in critical path 1.6 times the unit transistor to get the optimized energy delay product.

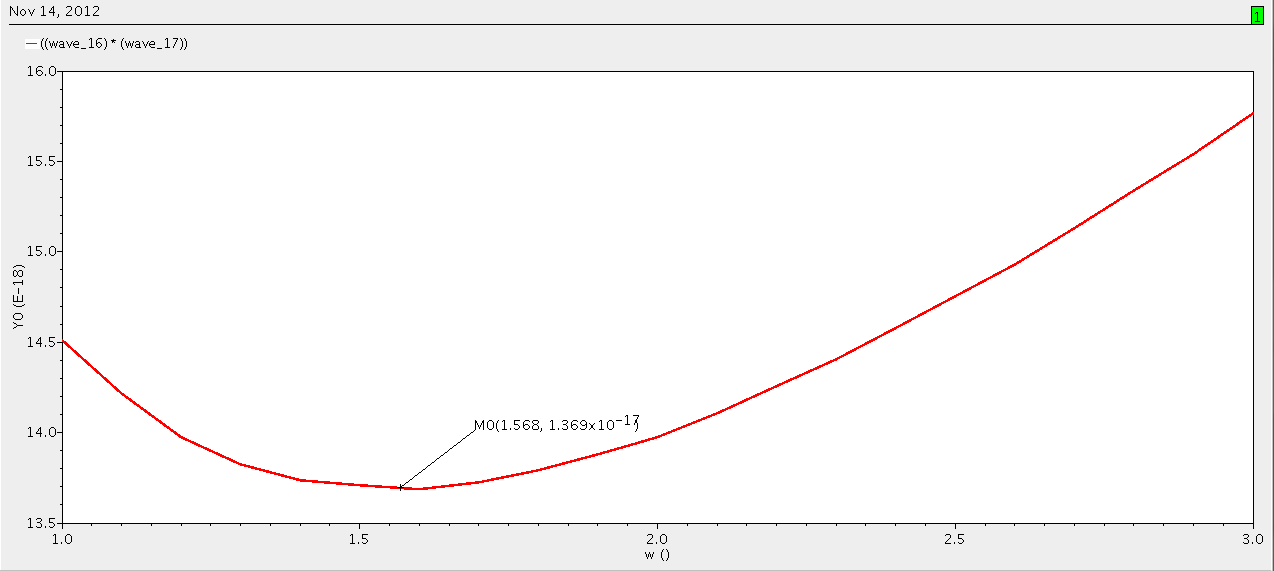


Fig. 10 Size of PMOS vs. Et product

1. Utilize the inversion property

Inversion property of carry ripple adder is another strategy we can use to optimize the delay of critical path. Table 1 shows the performance and power of a 4 bit CRA with and without inversion implementation.

|  |  |  |
| --- | --- | --- |
|  | Without Inversion | With inversion |
| tplh | 102.7ps | 103.2ps |
| tphl | 121.2ps | 103.6ps |
| tp | 111.95ps | 103.4ps |
| Power |  |  |

1. Transmission Gate Adder implementation

To further optimize the addition path, we found a transmission gate adder which can decrease power and performance a lot. This kind of adder has some drawbacks, like driving ability, etc.

1. Power simulation

Table 1 Power Simulation with Truncation

Conventional 8 bit by 8 bit array multiplier: Delay: 307.2 ps Power: 26.88 uW

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # of blocks turned off | Power(uW vd=1.1) | Power(vdd=1) | Power(vdd=0.9) | Power(vdd=0.8) | delay(ps) |
| 0 | 37.15 | 36.94 | 39.82 | 54.23 | 398 |
| 1 | 33.12 | 32.84 | 35.7 | 50.1 | 398 |
| 2 | 27.6 | 27.37 | 29.62 | 41.46 | 398 |
| 3 | 27.4 | 27.15 | 29.42 | 41.26 | 398 |
| 4 | 21.3 | 21.16 | 22.83 | 32.07 | 398 |
| 5 | 21.13 | 20.97 | 22.64 | 31.88 | 398 |
| 6 | 20.93 | 20.77 | 22.42 | 31.09 | 398 |

As we can see from the table, at 0 blocks turned off, our design consumes much more power than the conventional one. However, when more than 3 blocks are turned off we start to gain in power consumption. We can also see that, when we lower vdd to 1V we reduced a fraction of power, however, when the vdd lowered to below 0.9v, the power consumption goes up in which case a level shifter is needed.

1. Power gating simulation

Table 2 Power Gating Simulation

|  |  |  |
| --- | --- | --- |
| **Multiplier Row** | **Power(uW)** | **Delay(pS)** |
| No Power Gating | 9.94 | 59.5 |
| With Power Gating( | 4.01 | 46.67 |
| With Power Gating | 0.1 | N/A |

1. Error Characteristics Simulation:

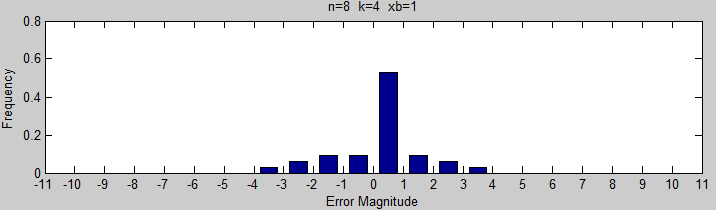


Figure 1 8 bits 4 blocks and 1 small block truncation

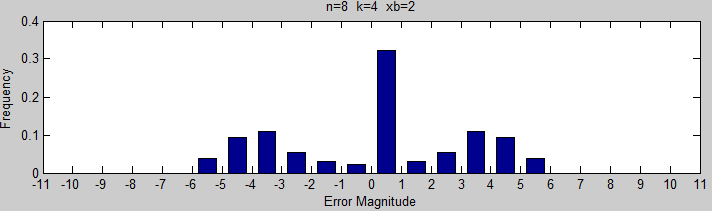


Figure 2 8 bits 4 blocks and 2 small block truncation

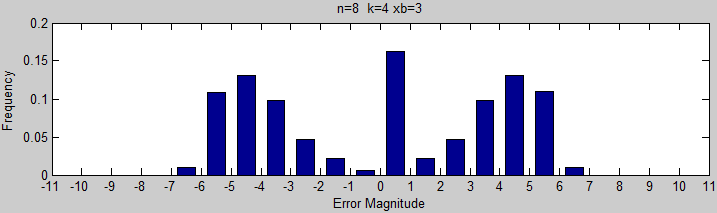


Figure 3 8 bits 4 blocks and 3 small block truncation

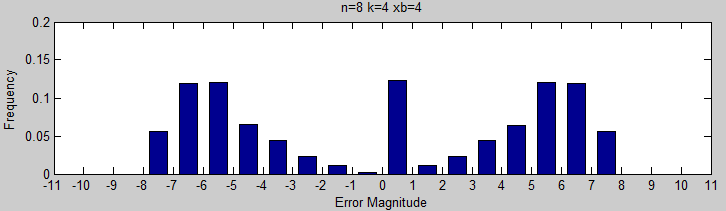


Figure 4 8 bits 4 blocks and 4 small block truncation

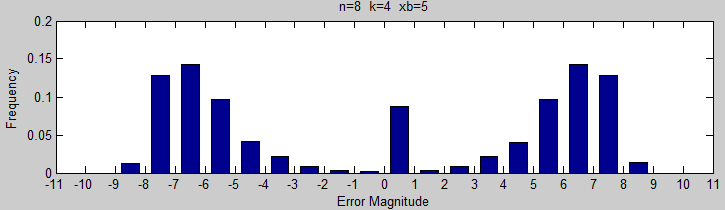


Figure 5 8 bits 4 blocks and 5 small block truncation

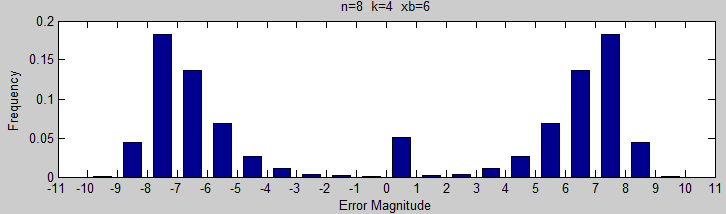


Figure 6 8 bits 4 blocks and 6 small block truncation

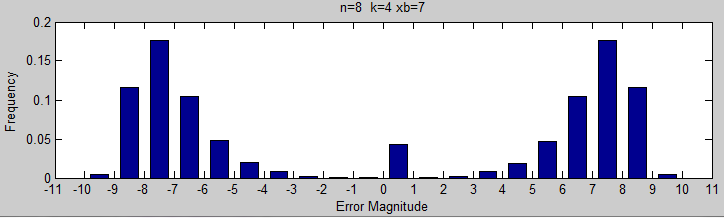


Figure 7 8 bits 4 blocks and 7 small block truncation

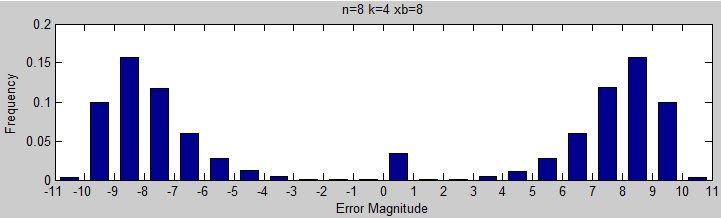


Figure 8 8 bits 4 blocks and 8 small block truncation

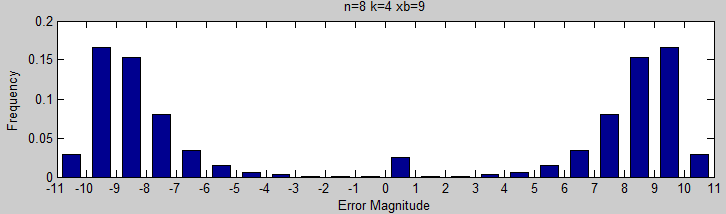


Figure 9 8 bits 4 blocks and 9 small block truncation

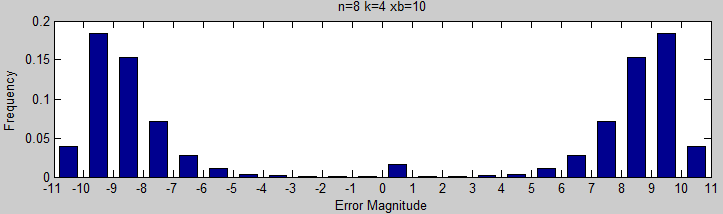


Figure 10 8 bits 4 blocks and 10 small block truncation

The X-axis represents the error magnitude in power of 2. For example, 1 on the positive x axis means an error magnitude of 2^1, and a 10 means 2^10. A -10 means -2^10. The y axis represents the error frequency, therefore the sum of all bars will equal to one. The series of histograms shows that as the number of blocks turned off goes up, the error characteristic of the multiplier shifts from frequent small magnitude error to frequent large magnitude error.

1. Current Progress:

Currently the 8 by 8 bit fixed point power aware block array imprecise multiplier has been successfully implemented in both algorithm level in python, java, and C and the transistor level in cadence. In the algorithm level, we have done 2million random input simulation for each individual parameter settings with a total of 40 of them. After the simulation, we analyzed the error characteristics of the circuits and plotted the error frequency VS. error magnitude histograms. It can be concluded that our design of the multiplier is a frequent small magnitude imprecise multiplier with small truncations. But the error magnitude and frequency goes up when the number of blocks also goes up. The functional characteristics of the multiplier in the both level have been simulated and verified. We also integrated the zero detection circuits and power gating circuits with the multiplier. Therefore, the multiplier now can turn off all zero blocks by detecting zero operands and turn off the corresponding small multipliers using power gating techniques. We also applied voltage scaling to the multiplier so that the non-critical path will have lower voltage while the critical path will have higher voltage. The voltage scaling does give some power savings. Primitive simulation results show that with full precision, our design is slightly slower and consumes more power. However, with the integration of zero detection and truncation circuits, the power consumption drops significantly.

1. Remaining tasks:
2. Reduce glitches in the design by implementing dynamic logic
3. Power optimization with inversion full adder
4. Random input vectors power simulation in cadence using Hsim
5. Implement imprecise final stage adder(almost correct adder)
6. Case study on JPEG compression
7. Try implementing dynamic logic to reduce glitches
8. Use ULP to quantify errors
9. Challenges:

There were a lot of challenges during the design and implementation of the power aware block array multiplier. For example, the original algorithm written in python would take more than 2weeks to finish all the simulation, therefore it is necessary to rewrite the algorithm in JAVA and parallelize the algorithm as much as possible to speed up the simulation process. It turns out that in JAVA, it only takes around 4 days to finishes all 50 set 2million random number simulation. At this stage, the algorithm has been implemented in C in order to do a case study on JPEG compression. And it should give faster performance in C.

Another challenge we faced was to figure out the critical path for our multiplier. Different from the conventional array multiplier which has multiple critical paths, our design has a critical path that’s somehow tricky in order to find a input vector that is also the critical path for the conventional multiplier.

In addition, we found out that there are some problems when we were trying to scale down the supply voltage of the non-critical path. At first, when we scale down a little bit of voltage, the power consumption drops. However, when we further scales down to 0.8V, the power consumption actually goes up. In this case, we found out that we need level shifters which could add some power consumption and delays to the circuits.

For Voltage Scaling, we will run the situation low voltage drives high voltage, so we need to utilize level shifter, which brings more overhead. It is a bottleneck now. We need to make a good argument about power and precision to compete with all other existing State of art multiplier. We are still exploring if we can use dynamic logic to minimize the glitches.

1. Design timeline and goals:

Originally we planned to not only implement the 8 by 8 bit multiplier but also the 16 bit and 32 bit multiplier. However, due to our misjudge on the difficulty of implementing the 8 by 8 bit multiplier, we may not have enough time to also implement the 16bit or the 32 bit multiplier. Therefore, in the remaining time frame of this semester, we will continue optimizing our design on the 8 by 8 bit multiplier. And the 16 bit and 32 bit can be done after this semester as a small research project. We expect 16 bit and 32 bit design will take much less time after we implemented the 8 by 8 bit multiplier.

1. Contributions:

Xinfei Guo:

1. VHDL implementation and simulation of 8 by 8 bit PABAIM
2. Implementation of 8 by 8 PABAIM in cadence
3. Test verification of each design
4. Power and Performance simulation in cadence
5. Zero detection and precision control circuit implementation in Cadence
6. Collaboration on final presentation preparation
7. Collaboration on final project report

Eric Zhang:

1. Implementation of algorithms in Python, JAVA, and C
2. 2 million random input vectors for 50 different parameter set
3. Design and analysis of the algorithm for the multiplier
4. Error characteristics analysis on the multiplier.
5. Power and delay simulation with full precision
6. Truncation power and delay simulation
7. Transmission gate adder integration
8. HSim simulation in progress